Deep and Fast: Deep Learning Hashing with Semi-supervised Graph Construction

Jingkuan Song, Lianli Gao, Fuhao Zou, Yan Yan

PII: S0262-8856(16)30011-7
DOI: doi: 10.1016/j.imavis.2016.02.005
Reference: IMAVIS 3466

To appear in: *Image and Vision Computing*

Received date: 15 September 2015
Revised date: 13 January 2016
Accepted date: 18 February 2016

Please cite this article as: Jingkuan Song, Lianli Gao, Fuhao Zou, Yan Yan, Deep and Fast: Deep Learning Hashing with Semi-supervised Graph Construction, *Image and Vision Computing* (2016), doi: 10.1016/j.imavis.2016.02.005

Deep and Fast: Deep Learning Hashing with Semi-supervised Graph Construction

Jingkuan Song[a,1], Lianli Gao[b], Fuhao Zou[c], Yan Yan[a]

[a] *University of Trento, Italy*
[b] *University of Electronic Science and Technology of China, China*
[c] *Huazhong University of Science and Technology, China*

Abstract

Learning-based hashing methods are becoming the mainstream for approximate scalable multimedia retrieval. They consist of two main components: hash codes learning for training data and hash functions learning for new data points. Tremendous efforts have been devoted to designing novel methods for these two components, i.e., supervised and unsupervised methods for learning hash codes, and different models for inferring hashing functions. However, there is little work integrating supervised and unsupervised hash codes learning into a single framework. Moreover, the hash function learning component is usually based on hand-crafted visual features extracted from the training images. The performance of a content-based image retrieval system crucially depends on the feature representation and such hand-crafted visual features may degrade the accuracy of the hash functions. In this paper, we propose a semi-supervised deep learning hashing (DLH) method for fast multimedia retrieval. More specifically, in the first component, we utilize both visual and label information to learn an optimal similarity graph that can more precisely encode the relationship among training data, and then generate the hash codes based on the graph. In the second stage, we apply a deep convolutional network to simultaneously learn a good multimedia representation and a set of hash functions. Extensive experiments on five popular datasets demonstrate the superiority of our DLH

*Corresponding author, jingkuan.song@unitn.it

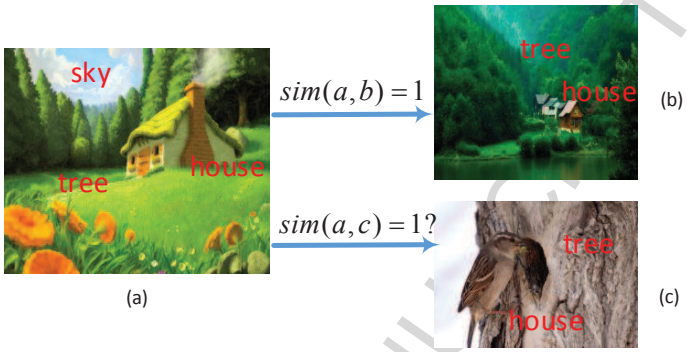


Figure 1: Purely using image labels to calculate the similarity between images is inaccurate. By using the widely-used similarity criteria stating that if two images have at least one common label, their similarity is defined as 1, $sim(a, b) = 1$ and $sim(a, c) = 1$. However, it is clear that (b) is closer to (a), both semantically and visually, compared with (c). Our main task is to learn a relative similarity between images by using visual and label information, based on which more accurate hash codes can be generated.

over both supervised and unsupervised hashing methods.

Keywords: Deep learning, Hashing, multimedia retrieval

1. Introduction

During the last decade, the amount of multimedia data has reached an unprecedented level, due to the development of multimedia and the Internet technologies. This has impacted a wide range of research areas, including multimedia retrieval, computer vision, database management, data mining, and social media analysis. A basic solution for dealing with massive multimedia databases is content-based image retrieval (CBIR), which aims to retrieve similar images from a database to the query image. Particularly, image retrieval has long been a major research theme due to many applications such as event detection, scene classification and face recognition [1, 2, 3].

Image retrieval is typically accomplished in three steps: feature extraction,

searching and post-processing. Over the last few years, a long stream of research efforts have been made to improve these three components [4]. From the perspective of image representation and search scheme, most of the successful scalable image retrieval algorithms can be categorized into: 1) quantized local features (e.g., BOW) [5, 6] indexed by an inverted index [7]; and 2) holistic features (e.g., GIST and HSV) indexed by compact hash codes [8, 9]. The BOW vector for an image is usually high-dimensional when used for large scale image search. In this case, the search efficiency results from the use of inverted lists, which speeds up the calculation of distances between sparse vectors. However, when the size of the image database reaches the million level, the speed degrades greatly. To scale to even larger databases, hashing methods are usually adopted. Each image is represented as a vector, either by extracting holistic feature directly or by compressing local descriptors into a vector representation using recent encoding techniques such as Fisher kernels [10] and the 'vector of locally aggregated descriptors' (VLAD) [11]. Then, images can be encoded by a small number of bytes using hashing methods [8, 12], but with the advantage of preserving some key properties inherited from visual features. For post-processing, after image retrieval returns a list of images, a potential re-ranking technique can be used to reorder the initial results. But this procedure is optional, and there is no well-accepted methods for post-processing due to the time limitation.

Although a variety of techniques have been proposed, image retrieval remains one of the most challenging problems. This is due to the fact that the global and local features have different strength and weakness, and specific search schemes have to be adopted based on different visual features. Therefore, there is an urgent need to find a more robust feature and an unified searching scheme for larger scale image retrieval. There arises several questions for scalable CBIR: 1) Is there a generic feature extraction method that is robust on different image datasets? 2) How to integrate this generic feature to existing searching schemes? Some recent efforts have been proposed to tackle these questions. In [8, 13], the authors propose utilizing supervised deep learning to learn image representation and hash codes for image retrieval and promising results were obtained. On

the other hand, Wan et al. [14] propose a general framework to apply deep learning to content-based image retrieval. However, these methods use the label information only to identify the similarity between two data points, which is inaccurate in a lot of cases. As shown in Fig. 1, image (b) is more similar than (c) to the query (a), both visually and semantically. If we only use the label information, (b) and (c) will have the same similarity to (a). Therefore, it is necessary to learn a more accurate similarity between images by using visual and label information.

In this paper, we propose integrating a generic feature into a hashing search scheme. More specifically, we present deep learning hashing (DLH) with semi-supervised graph construction to enhance the performance of hashing methods for scalable image retrieval. It is worth highlighting the following contributions: 1) We propose a framework to utilize the CNN-generated visual feature as a generic feature, and incorporate this feature into this hashing search scheme for efficient image retrieval on scalable datasets. 2) To infer the hash codes of the training set in both supervised and unsupervised settings, a novel semi-supervised graph construction approach is proposed and an efficient solution is devised. 3) We propose to simultaneously learn a good feature representation and a set of hash functions via CNN for new data points. (4) We conduct extensive experiments to exploit the performance of different hashing methods on five popular image datasets.

2. Related Work

To address the scalability issue in image retrieval, more recently, tremendous research efforts have been devoted to learning-based hashing methods [15, 16, 9, 12, 17, 18, 19, 20, 21, 22, 23], due to the compact binary representation and efficient Hamming distance. Such approaches map data points to compact binary codes through a hash function, which can be generally expressed as:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^L \tag{1}$$

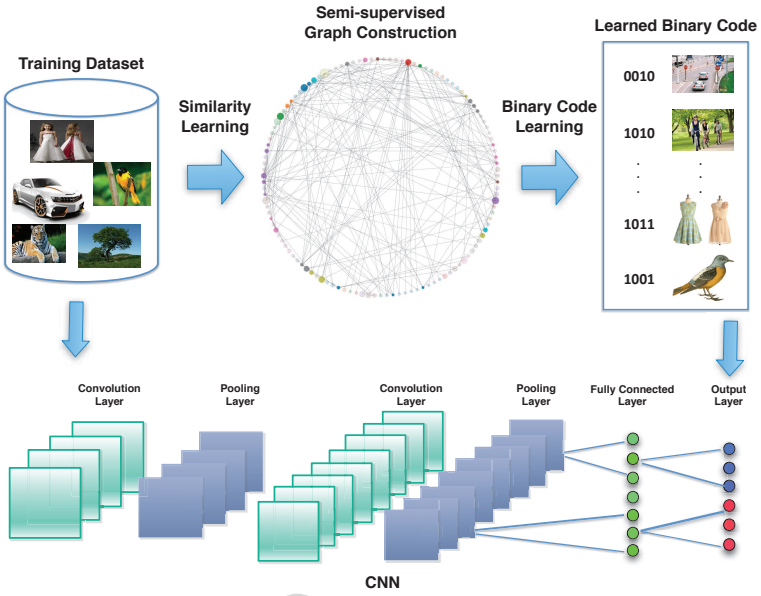


Figure 2: The overview of our proposed deep learning hashing with semi-supervised graph construction method.

where $\mathbf{x}$ is a m-dimensional vector, $\mathbf{h}(.)$ is the hash function, and $\mathbf{y}$ is a binary vector with code length L. The learning of the hash functions is usually based on the criterion of preserving some properties of the training data points. Typical approaches preserve code consistency property (*i.e.*, the similarity of binary codes should be consistent with that of the original data points) [15], similarity alignment property (*i.e.*, the Hamming distance of the binary code should approximate the Euclidean distance of the original data points) [24], order preserving property (i.e., the orders of a reference data item computed from the original space and the Hamming space should be aligned) [25], etc. However, these hashing methods are all based on hand-crafted visual features. On the other hand, we have witnessed dramatic progress in deep convolution networks in the last few years. Approaches based on deep networks have achieved state-of-the-art performance on image retrieval [14, 8], image classification [26], object detection [27] and other recognition tasks. The success of deep-networks-based methods for images is mainly due to their power to automatically learn effective

image representations. Xia *et al.* [8] developed a supervised hashing method for image retrieval, which simultaneously learns a good representation of images as well as a set of hash functions. Lai *et al.* [13] developed a 'one-stage' supervised hashing method for image retrieval, which generates bitwise hash codes for images via a carefully designed deep architecture. Zhao *et al.* [28] proposed employing multi-level semantic ranking supervision to learn deep hash functions based on CNN which preserves the semantic structure of multi-label images. However, all of the above CNN_based hashing methods are supervised learning approaches and cannot exploit the useful information from the unlabeled data samples.

3. Deep Learning Hashing with Semi-supervised Graph Construction

In this section, we introduce our method. In some supervised hashing methods, the learning task is formulated as a single optimization objective function which is complex and difficult to solve. While in other methods [29, 30], the learning process is decomposed into two stages: a hash code learning stage followed by a hash function inferring stage, where the corresponding optimization problems in each stage is relatively simple. Our DLH follows the two-stage paradigm. Fig. 2 gives an overview of the proposed DLH for CBIR. The framework consists of two components, namely hash codes learning via semi-supervised graph construction and hash functions inferring using CNN. Compared with [29, 30], DLH learns an optimal similarity graph for hash coding learning and utilizes CNN to learn an image representation and hashing function simultaneously.

3.1. Semi-supervised hash codes learning

Suppose $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ represents a set of N images, and $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$, where $\mathbf{f}_i = \{0, 1\}^C$ is the label for the i-th image ($1 \leq i \leq N$), and C is the number of classes. The task is to learn the hash codes $\mathbf{Y}$ based on $\mathbf{X}$ and $\mathbf{F}$. Suppose $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\}$, where $\mathbf{y}_i = \{-1, 1\}^L$ is the hash code for the i-th

training image ($1 \leq i \leq N$), and L is the code length of the hash code. For supervised hashing methods, a good criteria for $\mathbf{Y}$ is that the similarity of a pair of hash codes $\mathbf{y}_i$ and $\mathbf{y}_j$ should be consistent with the similarity of their labels, which can be formulated as:

$$\min_{\mathbf{Y}} \sum_{ij} \left(sim_{Sup}\left(\mathbf{f}_i, \mathbf{f}_j\right) - C \times sim_H\left(\mathbf{y}_i, \mathbf{y}_j\right)\right)^2 \tag{2}$$

where sim_{Sup} is usually defined as:

$$sim_{Sup}\left(\mathbf{f}_i, \mathbf{f}_j\right) = \begin{cases} +1, \ if \ \mathbf{f}_i \mathbf{f}_j^T > 0, \\ -1, \ else \end{cases} \tag{3}$$

and $\mathbf{f}_i$, $\mathbf{f}_j$ are the labels for the i-th and j-th images, and C is a constant.

For unsupervised hashing methods, a good criteria for $\mathbf{Y}$ is that the similarity of a pair of hash codes $\mathbf{y}_i$ and $\mathbf{y}_j$ should be consistent with their similarity in their original space, which can be formulated as:

$$\min_{\mathbf{Y}} \sum_{ij} \left(sim_{Uns}\left(\mathbf{x}_i, \mathbf{x}_j\right) - C \times sim_H\left(\mathbf{y}_i, \mathbf{y}_j\right)\right)^2 \tag{4}$$

where sim_{Uns} is usually defined as:

$$sim_{Uns}\left(x_i, \mathbf{x}_j\right) = \begin{cases} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / 2\sigma^2}, & if \ \mathbf{x}_i \in \mathcal{N}_K(\mathbf{x}_j) \\ & or \ \mathbf{x}_j \in \mathcal{N}_K(\mathbf{x}_i) \\ 0, \text{else} \end{cases} \tag{5}$$

where $\mathcal{N}_K(\cdot)$ is the K-nearest-neighbor set and $1 \leq (i, j) \leq N$.

For supervised hashing, the similarity between two data points is set to $+1$ or -1, which is too rigid. As is shown in Fig. 1, the image pair (a,b) has higher similarity than the pair (a,c), both visually and semantically. But they are assigned with the same value of similarity, which degrades the discriminative power of hashing. For unsupervised hashing methods, the variance σ will affect the performance significantly, and it is usually empirically tuned. In this work, we propose integrating supervised and unsupervised hashing into a single framework and automatically learning an optimal similarity graph by using both visual feature and image labels, which can more precisely encode the relationships between data points.

An optimal similarity graph S should be smooth on all these information cues, which can be formulated as:

$$\begin{aligned}&\min_{\mathbf{S},\mathbf{Y}} \mathbf{g}\left(\mathbf{F},\mathbf{S}\right)+\alpha\mathbf{h}\left(\mathbf{X},\mathbf{S}\right)+\beta\mathbf{r}\left(\mathbf{S}\right)\\&+\gamma\sum_{ij}\left(s_{ij}-C\times sim_H\left(\mathbf{y}_i,\mathbf{y}_j\right)\right)^2\end{aligned} \tag{6}$$

where $\mathbf{g}\left(\mathbf{F},\mathbf{S}\right)$ is the penalty function to measure the smoothness of $\mathbf{S}$ on the label information $\mathbf{F}$ and $\mathbf{h}\left(\mathbf{X},\mathbf{S}\right)$ is the loss function to measure the smoothness of $\mathbf{S}$ on the feature $\mathbf{X}$. $\mathbf{r}\left(\mathbf{S}\right)$ are regularizers defined on the target $\mathbf{S}$. α, β and γ are balancing parameters.

The penalty function $\mathbf{g}\left(\mathbf{F},\mathbf{S}\right)$ should be defined in the way such that close labels have high similarity and vice versa. In this paper, we define it as follows:

$$\mathbf{g}\left(\mathbf{F},\mathbf{S}\right)=\sum_{ij}\|\mathbf{f}_i-\mathbf{f}_j\|_2^2 s_{ij} \tag{7}$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are the labels of data point $\mathbf{x}_i$ and $\mathbf{x}_j$. Similarly, $\mathbf{h}\left(\mathbf{X},\mathbf{S}\right)$ can be defined as:

$$\mathbf{h}\left(\mathbf{X},\mathbf{S}\right)=\sum_{ij}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2 s_{ij} \tag{8}$$

Instead of preserving all the pairwise distances, we consider preserving the pair distances of the K-nearest neighbors here, i.e., if $\mathbf{x}_i$ and $\mathbf{x}_j$ (or f_i *and* f_j) are not K-nearest neighbors of each other, their distance will be set to a large constant. The regularizer term $\mathbf{r}\left(\mathbf{S}\right)$ is defined as:

$$\mathbf{r}\left(\mathbf{S}\right)=\|\mathbf{S}\|_F^2 \tag{9}$$

We define

$$sim_H\left(\mathbf{y}_i,\mathbf{y}_j\right)=\frac{1}{L}\left(L-\|\mathbf{y}_i,\mathbf{y}_j\|_H\right)=\frac{1}{2L}\mathbf{y}_i\mathbf{y}_j^T+\frac{1}{2} \tag{10}$$

and we further constraint that $\mathbf{S}\geq 0$ and $\mathbf{S1}=\mathbf{1}$. Then we can obtain the objective function for learning hash codes by replacing $\mathbf{g}\left(\mathbf{F},\mathbf{S}\right)$, $\mathbf{h}\left(\mathbf{X}^t,\mathbf{S}\right)$, $\mathbf{r}\left(\mathbf{S},\alpha\right)$

and $sim_H\left(\mathbf{y}_i, \mathbf{y}_j\right)$ in (6) using (7), (8), (9) and (10):

$$\begin{aligned} &\min_{\mathbf{S},\mathbf{Y}} \left(\sum_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 s_{ij} + \alpha \sum_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 s_{ij} \right. \\ &\left. +\beta \sum_{ij} \left(s_{ij} - \frac{C}{2L} \times \mathbf{y}_i \mathbf{y}_j^T - \frac{C}{2}\right)^2 + \gamma \|\mathbf{S}\|_F^2 \right) \\ &s.t. \begin{cases} \mathbf{S} \geq 0, \mathbf{S1} = \mathbf{1} \\ \mathbf{Y} \in \{-1, 1\}^{N \times L} \end{cases} \end{aligned} \tag{11}$$

3.2. Iterative optimization

We propose an iterative method to minimize the above objective function (11). Once these initial values are given, we iteratively update $\mathbf{S}$ and $\mathbf{Y}$. These steps are described as below.

3.2.1. Update S

By fixing $\mathbf{Y}$, we can obtain $\mathbf{S}$ by optimizing (11). It is equivalent to optimize the following objective function:

$$\begin{aligned} &\min_{\mathbf{S}} \sum_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 s_{ij} + \alpha \sum_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 s_{ij} \\ &+\beta \sum_{ij} \left(s_{ij} - \frac{C}{2L} \times \mathbf{y}_i \mathbf{y}_j^T - \frac{C}{2}\right)^2 + \gamma \|\mathbf{S}\|_F^2 \\ &= \min_{\mathbf{S}} \sum_i t\mathbf{r}\left(\left(\mathbf{a}_i + \alpha \mathbf{b}_i\right) \mathbf{s}_i{}^T + \gamma \mathbf{s}_i \mathbf{s}_i{}^T\right) + \beta \|\mathbf{s}_i - \mathbf{d}_i\|^2 \\ &s.t. \mathbf{S} \geq 0, \mathbf{S1} = \mathbf{1} \end{aligned} \tag{12}$$

where $\mathbf{a}_i = \{a_{ij}, 1 \leq j \leq N\}$ with $a_{ij} = \|\mathbf{f}_i - \mathbf{f}_j\|_2^2$, $\mathbf{b}_i = \{b_{ij}, 1 \leq j \leq N\}$ with $b_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ and $\mathbf{d}_i = \{d_{ij}, 1 \leq j \leq N\} \in R^{1 \times N}$ with $d_{ij} = \frac{C}{2L} \times \mathbf{y}_i \mathbf{y}_j^T + \frac{C}{2}$.

Since $\mathbf{d}_i$ is a constant, (12) can be reformulated as:

$$\begin{aligned} &\min_{\mathbf{S}, \mathbf{S} \geq 0, \mathbf{S1}=\mathbf{1}} \sum_i t\mathbf{r}\left(\left(\beta + \gamma\right) \mathbf{s}_i \mathbf{s}_i{}^T + \left(\mathbf{a}_i + \alpha \mathbf{b}_i - 2\beta \mathbf{d}_i\right) \mathbf{s}_i{}^T\right) \\ &\Rightarrow \min_{\mathbf{S}, \mathbf{S} \geq 0, \mathbf{S1}=\mathbf{1}} \sum_i tr\left(\mathbf{s}_i \mathbf{s}_i{}^T + \frac{\mathbf{a}_i + \alpha \mathbf{b}_i - 2\beta \mathbf{d}_i}{2(\beta + \gamma)} \mathbf{s}_i{}^T\right) \end{aligned} \tag{13}$$

and it is equivalent to:

$$\min_{\mathbf{S}, \mathbf{S} \geq 0, \mathbf{S1}=\mathbf{1}} \sum_i \left\| \mathbf{s}_i + \frac{\mathbf{a}_i + \alpha \mathbf{b}_i - 2\beta \mathbf{d}_i}{2\left(\beta + \gamma\right)} \right\|_2^2 \tag{14}$$

The constraint in problem (14) is simplex. We use the accelerated projected gradient method to linearly solve this problem. The critical step of the projected gradient method is to solve the following proximal problem:

$$\min_{\mathbf{x}\geq 0,\mathbf{x}^T\mathbf{1}=1} \frac{1}{2}\|\mathbf{x}-\mathbf{c}\|_2^2 \tag{15}$$

We will introduce an efficient approach to solve this problem. We write the Lagrangian function of problem (15) as

$$\frac{1}{2}\|\mathbf{x}-\mathbf{c}\|_2^2 - \tau(\mathbf{x}^T 1 - 1) - \rho^T\mathbf{x}, \tag{16}$$

where τ and ρ are Lagrangian coefficients. Suppose the optimal solution to the proximal problem (15) is $\mathbf{x}^*$, the associated Lagrangian coefficients are τ^* and ρ^*. Then according to the KKT condition, we have the following equations:

$$\begin{cases} \forall i, \quad x_i^* - c_i - \tau^* - \rho_i^* = 0 & (17) \\ \forall i, \quad x_i^* \geq 0 & (18) \\ \forall i, \quad \rho_i^* \geq 0 & (19) \\ \forall i, \quad x_i^*\rho_i^* = 0 & (20) \end{cases}$$

Eq.(17) can be written as $\mathbf{x}^* - c - \tau^* 1 - \rho^* = 0$. According to the constraint $1^T\mathbf{x}^* = 1$, we have $\tau^* = \frac{1-1^Tc-1^T\rho^*}{n}$. So $\mathbf{x}^* = (c - \frac{11^T}{n}c + \frac{1}{n}1 - \frac{1^T\rho^*}{n}1) + \rho^*$.

Denote $\bar{\rho}^* = \frac{1^T\rho^*}{n}$ and $\mathbf{u} = \mathbf{c} - \frac{11^T}{n}\mathbf{c} + \frac{1}{n}1$, then we can write $\mathbf{x}^* = \mathbf{u} + \rho^* - \bar{\rho}^*\mathbf{1}$. So $\forall i$ we have

$$x_i^* = u_i + \rho_i^* - \bar{\rho}^*. \tag{21}$$

According to Eq.(18)-(21) we know $u_i + \rho_i^* - \bar{\rho}^* = (u_i - \bar{\rho}^*)_+$. Then we have

$$x_i^* = (u_i - \bar{\rho}^*)_+. \tag{22}$$

So we can obtain the optimal solution $\mathbf{x}^*$ if we know $\bar{\rho}^*$.

We write Eq.(21) as $\rho_i^* = x_i^* + \bar{\rho}^* - u_i$. Similarly, according to Eq.(18)-(20) we know $\rho_i^* = (\bar{\rho}^* - u_i)_+$. Suppose $\mathbf{c}$ is a m-dimensional vector, then we have $\bar{\rho}^* = \frac{1}{m}\sum_{i=1}^{m}(\bar{\rho}^* - u_i)_+$. Defining a function as:

$$f(\bar{\rho}) = \frac{1}{n}\sum_{i=1}^{n}(\bar{\rho} - u_i)_+ - \bar{\rho}, \tag{23}$$

so $\mathbf{f}(\bar{\rho}^*) = 0$ and we can solve the root finding problem with the Newton method to obtain $\bar{\rho}^*$.

Then each $\mathbf{s}_i$ can be efficiently solved, and we will get the updated similarity graph $\mathbf{S}$.

3.2.2. Update $\boldsymbol{Y}$

By fixing $\mathbf{S}$, we can obtain $\mathbf{Y}$ by optimizing (11). It is equivalent to optimize the following objective function:

$$\begin{aligned} &\min_{\mathbf{Y}} \sum_{ij} \left(\tfrac{2L}{C} \left(s_{ij} - \tfrac{C}{2} \right) - \mathbf{y}_i \mathbf{y}_j{}^T \right)^2 \\ &\Rightarrow \min_{\mathbf{Y}} \left\| \bar{\mathbf{S}} - \mathbf{Y}\mathbf{Y}^T \right\|_F^2 \\ &s.t. \mathbf{Y} \in \{-1, 1\}^L \end{aligned} \tag{24}$$

This algorithm sequentially or randomly chooses one entry in Y to update while keeping other entries fixed. Let $\mathbf{Y} = [\mathbf{y}_{.1}, ..., \mathbf{y}_{.L}]$, where $\mathbf{y}_{.p} \in \mathbb{R}^{n \times 1}$ is the p-th column of $\mathbf{Y}$ (the p-th bit of all data). The objective function (24) can be rewritten as:

$$\begin{aligned} &\min_{\mathbf{Y}} \left\| \bar{\mathbf{S}} - \sum_{l \neq p} \mathbf{y}_{.l} \mathbf{y}_{.l}^T - \mathbf{y}_{.p} \mathbf{y}_{.p}^T \right\|_F^2 \\ &= \min_{\mathbf{Y}} \left\| \mathbf{E} - \mathbf{y}_{.\mathbf{p}} \mathbf{y}_{.\mathbf{p}}^{\mathbf{T}} \right\|_F^2 = \min_{\mathbf{Y}} \sum_{i,j}^{n} \left(e_{ij} - y_{ip} y_{jp} \right)^2 \end{aligned} \tag{25}$$

where $\mathbf{E} = \bar{\mathbf{S}} - \sum_{l \neq p} \mathbf{y}_{.l} \mathbf{y}_{.l}^T$ and $\overline{s_{ij}} = \frac{2L}{C} \left(s_{ij} - \frac{C}{2} \right)$. Suppose we want to update y_{qp} by fixing the other hash codes. Optimizing (24) is equivalent to:

$$\begin{aligned} &\min_{y_{qp}} \sum_{i,j}^{n} \left(e_{ij} - y_{ip} y_{jp} \right)^2 \\ &= \min_{y_{qp}} \sum_{ij}^{n} \left(e_{ij} \times e_{ij} - 2 e_{ij} y_{ip} y_{jp} + y_{ip} y_{jp} y_{jp} y_{ip} \right) \\ &= \min_{y_{qp}} \sum_{i,j}^{n} \left(-2 e_{ij} y_{ip} y_{jp} + const \right) \end{aligned} \tag{26}$$

and it is equivalent to:

$$\begin{aligned}
&\max_{y_{qp}} \left(e_{qq} y_{qp} y_{qp} + \sum_{j=1}^{n} e_{qj} y_{qp} y_{jp} + \sum_{i=1}^{n} a_{iq} y_{ip} y_{qp} \right) \\
&\Rightarrow \max_{y_{qp}} \left(\sum_{j=1, j\neq q}^{n} e_{qj} y_{jp} + \sum_{i=1, i\neq q}^{n} e_{iq} y_{ip} \right) y_{qp} \\
&\Rightarrow y_{qp}^{*} = \operatorname{sgn} \left(\sum_{j=1, j\neq q}^{n} e_{qj} y_{jp} + \sum_{i=1, i\neq q}^{n} e_{iq} y_{ip} \right)
\end{aligned} \tag{27}$$

We update $\mathbf{S}$ and $\mathbf{Y}$ iteratively until the objective function (11) converges.

3.3. Supervised hash functions learning via CNN

In this section, we introduce hash functions learning via CNN, which consists of two stages: (i) feature extraction using CNN, (ii) hash functions learning on the features. Following [26], we use Caffe [31] for the implementation of deep CNN learning.

In general, the deep convolutional network, as shown in Fig. 2, consists of convolution/maxpooling layers, and fully connected/output layers. Specifically, the first layer is the input layer which adopts the mean-centered raw RGB pixels in intensity value. The first and the second convolution layers are followed by a response normalization layers and a max pooling layers, while the third, fourth, and fifth convolution layers are connected to each other without any intervening pooling or normalization. Following the convolutional layers, there are two more fully connected layers and the output layer.

We make use of the generated hash codes and discrete class labels of the training images for training. Specifically, for n training images in c classes (an image may belong to multiple classes), we define a $n \times c$ discrete label matrix $\mathbf{F} \in \{0,1\}^{N \times C}$, where $\mathbf{F}_{ij} = 1$ if the i-th training image belongs to the j-th class, otherwise $\mathbf{F}_{ij} = 0$. For the output layer in our network, in addition to defining the L output units (the blue nodes in the output layer), we add c output units (the red nodes in the output layer in Fig. 2) which correspond to the class labels of the training images. By incorporating the image class labels as a part in the output layer, we enforce the network to learn a shared image representation

which matches both the approximate hash codes and the image labels. This can be regarded as a transfer learning case in which the incorporated image labels are expected to be helpful for learning a more accurate image representation (i.e. the hidden units in the fully connected layer). This image representation may be advantageous for hash functions learning. The convolutional network automatically learns the image representation (represented by the green nodes in the fully connected layer) as well as a set of hash functions (represented by the solid lines between the green nodes and the nodes in the last layer). In prediction, one can input a test image to the trained network and obtain the hash code from the values of the blue nodes in the output layer.

4. Experiments

We evaluate our algorithm on the task of high-dimensional approximate nearest neighbor (ANN) search. Firstly, we study the influence of the parameters in our algorithm. Then, we compare our results with state-of-the-art supervised and unsupervised hashing methods on five popular datasets.

4.1. Settings

We evaluate the proposed method on five benchmark datasets with different kinds of images. 1) AWA dataset consists of 30,475 images of 50 animals classes with six pre-extracted feature representations for each image. 2) CIFAR consists of 60,000 32×32 color tinny images which are categorized into 10 classes (6K images per class); 3) ESPGAME contains a wide variety of images including logos, drawings, and personal photos. Following [32], we use the subset of 20,000, out of the 60,000 images publicly available. 4) IAPR contains around $20,000$ images, including logos, drawings, and personal photos; 5) The NUS-WIDE dataset has nearly 270,000 images collected from the web. It is a multi-label dataset in which each image is annotated with one or multiple semantic tags (class labels) from 81 concept tags. For ESPGAME and IAPR, we use the features extracted by [33] and then reduce the dimensionality to 1000 by PCA. For the others, we utilize the features provided with the datasets.

The datasets are divided into training set, testing set and query set. For CIFAR, the provided split is adopted. For AWA and NUSWIDE, 10,000 data points are randomly selected for training and the rest for testing. For ESPGAME and IAPR, 10,000 data points are selected for training and the whole datasets are used for testing. 1,000 queries are randomly selected from the training set.

ANN search is conducted to evaluate our proposed approaches, and two indicators are reported.

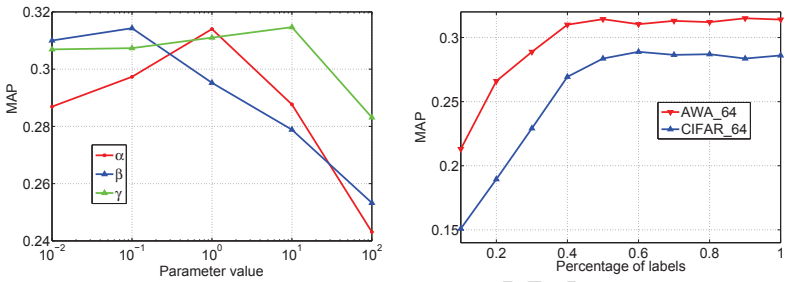
- Recall: the proportion over all queries where the true nearest neighbor falls within the top ranked vectors by the approximate distance.
- Mean Average Precision (MAP): For a single query, Average Precision (AP) is the average of the precision value obtained for the set of top-k results, and this value is then averaged over all the queries. The larger the MAP, the better the performance is.

We compare our SSH with other state-of-the-art supervised and unsupervised hashing algorithms, such as spectral hashing (SH) [15], iterative quantization (ITQ) hashing [34] and K-means Hashing (KMH) [12]. We also compare with two supervised hashing methods KSH [35] and FastH [29].

All the experiments were conducted on a Linux server with NVIDIA Tesla K20 GPUs. K20 is featuring 13 active SMXes along with 5 memory controllers and 1.25MB of L2 cache, attached to 5GB of GDDR5. In our experiments, the CNN model was trained on only one K20 GPU which has enough memory for our training tasks.

4.2. Parameters

There are several parameters, e.g., α, β, γ and the number of tagged training data, affecting the performance of our algorithm. In this subsection, we study the performance variation with different parameters. Due to the space limit, we only report the results on AWA dataset. The default settings for the parameters are: $\alpha = 1$, $\beta = 10^{-1}$, $\gamma = 10^{1}$ and 40% of the training data points are with labels.



(a) Performance variation with different α (b) Performance variation with different N

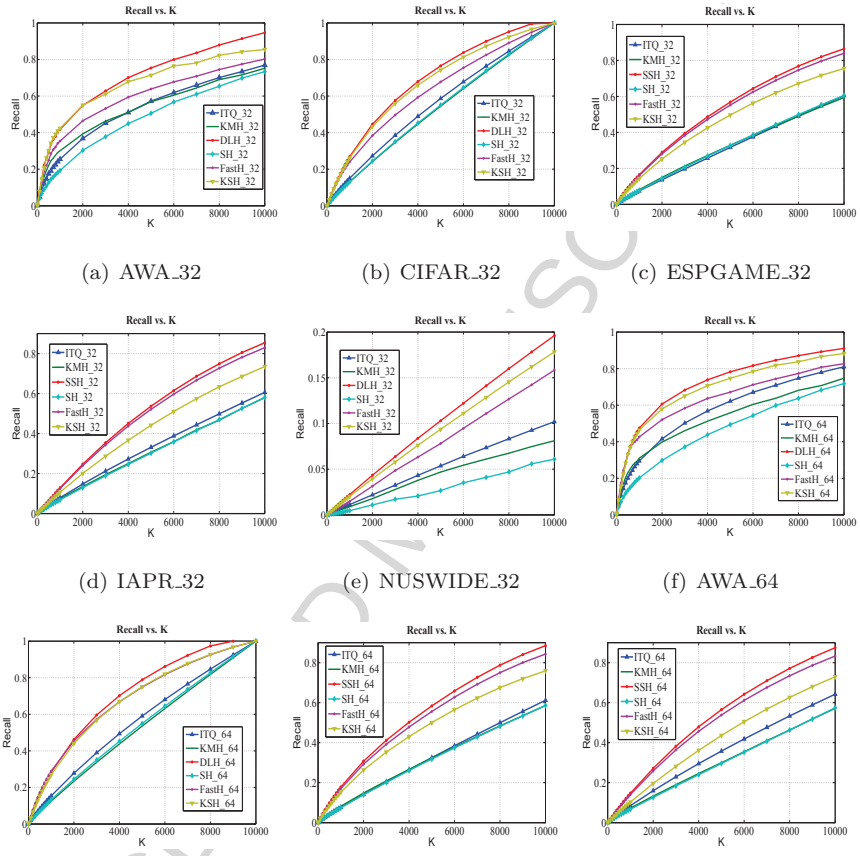
Figure 3: Parameters study with code length 32 and 64 on SIFT1M

We tune α, β and γ from $10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}$, and the results are shown in Fig. 3(a). When α is relatively large, e.g., $\alpha = 10^{2}$, the worst performance is achieved. That is to say, if the impact of visual features is dominant in constructing the optimal graph S, the MAP is unsatisfactory. With the increase of α from $\alpha = 10^{-2}$, the performance is slightly improved until reaching the peak at $\alpha = 10^{0}$. Further raising α will result in a slight drop of the performance, which means that balancing the partial tags and multiple features is better than using only one of them. On the other hand, the performance keeps stable when β is relative small, e.g., $\beta \leq 10^{-1}$, and it decreases quickly when β is set to a large value. Also, MAP is not very sensitive to γ when $\gamma \leq 10^{1}$.

The size of labeled training dataset affects the accuracy of the model. We use 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the training data as labeled data and the rest as unlabeled data. The performance variation is illustrated in Fig. 3(b). There is an increasing trend with the rising of the number of labeled data. When 40%-50% of the training data are with labels, further increasing the number of the labeled data will not improve the MAP significantly.

4.3. Results

Fig. 4(a)-4(i) show the comparison on the five datasets and Table 1 presents the MAP for the different hashing methods. We have tested L=32 and L=64.



(a) AWA_32 (b) CIFAR_32 (c) ESPGAME_32

(d) IAPR_32 (e) NUSWIDE_32 (f) AWA_64

(g) CIFAR_64 (h) ESPGAME_64 (i) IAPR_64

Figure 4: Comparison of different hashing methods on different datasets

From these figures, we have the following observations:

- Our method consistently outperforms the other hashing methods in all datasets. KSH and FastH are competitive in most settings. However, great improvement of SSH over FastH is achieved on AWA dataset. This indicates that a good image representation and the CNN-generated hash functions can improve the performance of hashing methods.

- With the increase of code length, the performance of different hashing methods is improved accordingly. Generally, MAP improvements of su-

Table 1: MAP of different methods with code length 32 and 64 on different databases

Methods	AWA	CIFAR	ESPGAME	IAPR	NUSWIDE
ITQ_32	13.9	13.9	5.63	9.56	7.78
KMH_32	18.0	11.8	5.82	8.96	5.86
SH_32	9.24	12.0	5.54	8.83	5.12
FastH_32	21.1	22.3	10.1	13.3	11.6
KSH_32	27.5	24.2	8.63	13.0	14.7
SSH_32	**28.3**	**25.1**	**11.4**	**14.6**	**15.2**
ITQ_64	16.4	14.1	5.82	10.3	7.91
KMH_64	19.1	11.5	5.83	8.96	6.94
SH_64	9.80	12.1	5.43	8.77	7.81
FastH_64	30.2	27.8	10.6	14.8	12.1
KSH_64	29.3	27.4	8.72	12.6	15.2
SSH_64	**31.4**	**28.6**	**11.5**	**15.6**	**17.3**

pervised hashing methods are more significant than that of unsupervised hashing methods. On ESPGAME and IAPR datasets, the performance even becomes worse when the code length increases.

- The supervised hashing methods outperform their unsupervised counterparts significantly. More specifically, the MAP gap between the best supervised and unsupervised hashing methods is 12%, 14%, 6%, 5% and 10% respectively, for AWA, CIFAR, ESPGAME, IAPRTC and NUSWIDE datasets, with $L = 64$. This indicates that exploiting the content information only cannot preserve the label-based similarity.

- For all the unsupervised methods, similar performance is achieved. In general, SH performs the worst while ITQ and KMH are comparable to each other.

5. Conclusions

In this paper, we propose a general framework for semi-supervised deep learning hashing (DLH) method for fast multimedia retrieval. In this framework, we firstly utilize both visual feature and class labels to learn an optimal similarity

graph to more precisely encode the relationship among training data. Based on this graph, the hash codes can be generated. Then, we apply the deep convolutional network to automatically learn a set of hash functions to get the hash codes for new data points. Extensive experiments on five popular datasets demonstrate the superiority of our DLH compared to both supervised and unsupervised hashing methods.

6. References

[1] R. Datta, D. Joshi, J. Li, J. Z. Wang, Image retrieval: Ideas, influences, and trends of the new age, ACM Computing Survey 40 (2).

[2] L. Zhang, L. Wang, W. Lin, Generalized biased discriminant analysis for content-based image retrieval, TSMCB 42 (1) (2012) 282–290.

[3] R. Cappelli, Fast and accurate fingerprint indexing based on ridge orientation and frequency, TSMCB 41 (6) (2011) 1511–1521.

[4] C. Böhm, S. Berchtold, D. A. Keim, Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases, ACM Computing Survey 33 (3) (2001) 322–373.

[5] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, C. Schmid, Aggregating local image descriptors into compact codes, IEEE Trans. Pattern Anal. Mach. Intell. 34 (9) (2012) 1704–1716.

[6] S. Zhang, M. Yang, T. Cour, K. Yu, D. N. Metaxas, Query specific fusion for image retrieval, in: ECCV, 2012.

[7] D. Nistér, H. Stewénius, Scalable recognition with a vocabulary tree, in: CVPR, 2006.

[8] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: AAAI, 2014.

[9] J. Song, Y. Yang, Z. Huang, H. T. Shen, R. Hong, Multiple feature hashing for real-time large scale near-duplicate video retrieval, in: ACM Multimedia, 2011.

[10] W. Zhou, Y. Lu, H. Li, Y. Song, Q. Tian, Spatial coding for large scale partial-duplicate web image search, in: ACM Multimedia, 2010.

[11] L. Zhang, J. Jiang, S. Cui, Achievable rate regions for discrete memoryless interference channel with state information, in: ICC, 2011.

[12] K. He, F. Wen, J. Sun, K-means hashing: An affinity-preserving quantization method for learning binary compact codes, in: CVPR, 2013.

[13] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: CVPR, 2015.

[14] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, J. Li, Deep learning for content-based image retrieval: A comprehensive study, in: ACM Multimedia, 2014.

[15] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008.

[16] J. Wang, O. Kumar, S.-F. Chang, Semi-supervised hashing for scalable image retrieval, in: CVPR, 2010.

[17] M. Norouzi, D. J. Fleet, Cartesian k-means, in: CVPR, 2013.

[18] J. Wang, H. T. Shen, J. Song, J. Ji, Hashing for similarity search: A survey, CoRR abs/1408.2927.

[19] X. Zhu, L. Zhang, Z. Huang, A sparse embedding and least variance encoding approach to hashing, IEEE Transactions on Image Processing 23 (9) (2014) 3737–3750.

[20] L. Liu, M. Yu, L. Shao, Multiview alignment hashing for efficient image search, TIP 24 (3) (2015) 956–966.

[21] M. Yu, L. Liu, L. Shao, Structure-preserving binary representations for rgb-d action recognition, TPAMI.

[22] L. Liu, M. Yu, L. Shao, Unsupervised local feature hashing for image similarity search, TCYB.

[23] L. Liu, L. Shao, Sequential compact code learning for unsupervised image hashing, TNNLS.

[24] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: NIPS, 2009.

[25] M. Norouzi, D. J. Fleet, Minimal loss hashing for compact binary codes, in: ICML, 2011.

[26] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS, 2012.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, CoRR abs/1409.4842.

[28] F. Zhao, Y. Huang, L. Wang, T. Tan, Deep semantic ranking based hashing for multi-label image retrieval, in: CVPR, 2015.

[29] G. Lin, C. Shen, Q. Shi, A. van den Hengel, D. Suter, Fast supervised hashing with decision trees for high-dimensional data, in: CVPR, 2014.

[30] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: SIGIR, 2010.

[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093.

[32] M. Chen, A. Zheng, K. Q. Weinberger, Fast image tagging, in: ICML, 2013.

[33] M. Guillaumin, T. Mensink, J. J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: ICCV, 2009.

[34] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. PAMI 35 (12) (2013) 2916–2929.

[35] W. Liu, J. Wang, R. Ji, Y. Jiang, S. Chang, Supervised hashing with kernels, in: CVPR, 2012.

Highlights

- A semi-supervised graph approach is proposed to infer the hash codes.
- We simultaneously learn visual features and hash functions.
- Extensive experiments are conducted to exploit the performance of our method.